APPENDIX C

```
/* ----------------------------------------------------------------------------
   Copyright (c) Masimo Corporation (tm) 1992, 1993. All Rights Reserved.

   File: smanc.c1

   Description: Improved Normalized Least Squares Lattice ANC

   Public Functions:    SANC_Calc
                        SANC_Init

   Notes:
        This version uses many of the same optimization techniques as the
        .asm version.

   History:
        HGK     04/29/93        Design Note SDN43 Rev A
   ------------------------------------------------------------------------- */

#define MODULE_ID       1007

#include <masimo.h>     /* platform descriptions        */
#include <math.h>

#include <smanc.h>      /* self                         */

#define MAX(a,b) (a) > (b) ? (a) : (b)
#define MIN(a,b) (a) < (b) ? (a) : (b)

#define MIN_VAL     0.01
#define MAX_DEL     0.9999999999999999
#define MIN_DEL    -0.9999999999999999
#define MAX_RHO     2.0
#define MIN_RHO    -2.0
#define MAX_BSERR   1.0
#define MIN_BSERR   1E-15

/* The following macros provide efficient access to the lattice */

#define xBERR     0
#define xBERR_1   1
#define xDELTA    2
#define xDELTA_1  3
#define xGAMMA    4
#define xGAMMA_1  5
#define xBSERR    6
#define xBSERR_1  7
#define xERR      8
#define xFERR     9
#define xRho     10

#define berr            (*(p + xBERR))
#define P_berr_1        (*(p + xBERR_1 - SANC_CELL_SIZE))
#define P_berr          (*(p + xBERR   - SANC_CELL_SIZE))
#define berr_1          (*(p + xBERR_1))

#define Bserr           (*(p + xBSERR))
#define Bserr_1         (*(p + xBSERR_1))
#define P_Bserr_1       (*(p + xBSERR_1 - SANC_CELL_SIZE))

#define P_delta         (*(p + xDELTA - SANC_CELL_SIZE))
```

```
#define delta               (*(p + xDELTA))
#define delta_1             (*(p + xDELTA_1))
#define P_delta_1           (*(p + xDELTA_1 - SANC_CELL_SIZE))

#define err                 (*(p + xERR))
#define N_err               (*(p + xERR + SANC_CELL_SIZE))

#define P_ferr              (*(p + xFERR - SANC_CELL_SIZE))
#define ferr                (*(p + xFERR))

#define gamma               (*(p + xGAMMA))
#define P_gamma             (*(p + xGAMMA   - SANC_CELL_SIZE))
#define N_gamma             (*(p + xGAMMA   + SANC_CELL_SIZE))
#define P_gamma_1           (*(p + xGAMMA_1 - SANC_CELL_SIZE))
#define gamma_1             (*(p + xGAMMA_1))

#define rho                 (*(p + xRho))


FLOAT32
SANC_Calc(
    SANC_DATA *anc,     /* input, context handle        */
    FLOAT32  nps,      /* input, noise plus signal     */
    FLOAT32  noise)    /* input, noise reference       */
{
    INT32        m;
    FLOAT32      *p;
    FLOAT32      B,F,B2,F2;
    FLOAT32      qd2,qd3;
    INT32        output_cell;
    BOOL         Bflag;

    BUG1(anc);   BUG1(nps); BUG1(noise);

    /* Update time delay elements in cell structure --------------- */

    p = (FLOAT32 *)anc->cells;
    for (m = 0; m <= anc->cc; m++) {
       gamma_1 = gamma;
       berr_1  = berr;
       Bserr_1 = Bserr;
       delta_1 = delta;
       p += SANC_CELL_SIZE;
    }

    /* Handle Cell # 0 ---------------------------------------------- */
    p = (FLOAT32 *)anc->cells;
    Bserr = anc->lambda * Bserr_1 + noise * noise;
    Bserr = MAX(Bserr, MIN_BSERR);

    ferr  = noise / SQRTF(Bserr);
    ferr  = MAX(ferr, MIN_DEL);
    ferr  = MIN(ferr, MAX_DEL);

    berr  = ferr;

    rho = anc->lambda * SQRTF(Bserr_1 / Bserr) * rho + berr * nps;

    N_err = nps - rho * berr;
    /* Initialize Cell v  r ------------------------------------------*/

    output_cell = anc->cc - 1;      /* Assume last cell for starter */
    Bflag = FALSE;

    for (m = 1; m < anc->cc; m++) {
        p += SANC_CELL_SIZE;

        B   = SQRTF(1.0 - P_berr_1 * P_berr_1);     B2 = 1.0/B;
        F   = SQRTF(1.0 - P_ferr   * P_ferr  );     F2 = 1.0/F;

        P_delta = P_delta_1 * F * B + P_berr_1 * P_ferr;
        P_delta = MAX(P_delta, MIN_DEL);
        P_delta = MIN(P_delta, MAX_DEL);
        qd3    = 1.0 - P_delta * P_delta;
        qd2    = 1.0 / SQRTF(qd3);

        ferr   = (P_ferr    - P_delta * P_berr_1) * qd2 * B2;
        ferr   = MAX(ferr, MIN_DEL);
        ferr   = MIN(ferr, MAX_DEL);

        berr   = (P_berr_1  - P_delta * P_ferr   ) * qd2 * F2;
        berr   = MAX(berr, MIN_DEL);
        berr   = MIN(berr, MAX_DEL);

        gamma  = P_gamma * (1.0 - P_berr * P_berr);
        gamma  = MAX(gamma, MIN_VAL);
        gamma  = MIN(gamma, MAX_DEL);

        Bserr  = P_Bserr_1 * qd3;

        /* update cell voter ---------------------------------- */
        if(Bserr < anc->voter && Bflag == FALSE) {
            output_cell = m;
            Bflag = TRUE;
        }

        Bserr   = MAX(Bserr, MIN_BSERR);

        rho   *= anc->lambda * SQRTF((Bserr_1 / Bserr) * (gamma / gamma_1));
        rho   += berr * err;
        rho    = MAX(rho, MIN_RHO);
        rho    = MIN(rho, MAX_RHO);

        N_err  = err - rho * berr;
    }

    p = (FLOAT32 *)&(anc->cells[output_cell /* *ANC_CELL_SIZE */]);
    return(N_err);


/OID
;ANC_Init(
    SANC_DATA    *anc)        /* input, context pointer        */

    FLOAT32      *p;
    INT32        m;

    BUG1(anc);
    p = (FLOAT32 *)anc->cells;
    for (m = 0; m <= anc->cc; m++) {
        rho     = 0.0;
        err     = 0.0;
        ferr    = 0.0;
        berr    = 0.0;
        berr_1  = 0.0;
        delta   = 0.0;
        delta_1 = 0.0;
        Bserr   = anc->min_error;
        Bserr_1 = anc->min_error;
        gamma   = MIN_VAL;
        gamma_1 = MIN_VAL;
        p      += SANC_CELL_SIZE;
    }
    p = (FLOAT32 *)anc->cells;            /* Cell # 0 special case     */
    gamma   = 1.0;
    gamma_1 = 1.0;
}
```